Université Hassan I

Faculté Des Sciences Et Techniques

Settat

La Catégorisation Automatique Des Questions De Stackoverflow

Pr : S. EL KAFHALI

Réalisé par : -MZAOUIR Imane

-ELGHZAOUI Ikram

-CHHAIB Fatima

Sommaire

Introduction Générale :

Stack Overflow est un site web de questions et réponses axé sur la programmation informatique et les technologies connexes. Il sert de plateforme pour les développeurs, les ingénieurs logiciels, les programmeurs et d'autres professionnels de l'informatique pour poser des questions, trouver des réponses, résoudre des problèmes et partager leurs connaissances.

Le site **Stack Overflow** fonctionne sur un modèle collaboratif où les utilisateurs peuvent poser des questions dans différents domaines liés à la programmation, tels que les langages de programmation, les Frameworks, les bibliothèques, les systèmes d'exploitation, les bases de données, les réseaux et bien d'autres encore. Les autres membres de la communauté peuvent ensuite répondre à ces questions, offrant ainsi des solutions, des conseils et des exemples de code.

Les tags les plus populaires incluent les langages de programmation et de script comme :

- JavaScript.
- Python.
- Java.
- C#.
- HTML.
- CSS.

Objectif :

Le but de ce projet est de créer un système qui propose automatiquement des tags lorsqu'un utilisateur pose une nouvelle question sur Stack Overflow.

Notre étude comporte trois grandes parties :

- Pre-processing des questions : pour la partie prétraitement des données, nous traiterons les données en supprimant les balises html courantes, la ponctuation, les mots alphanumériques et les mots vides.

On va rajouter aussi, une étape de Lemmatisation en faisant filtrer les STOPWORDS

- Extraction des topics : En faisant recours à la méthode non supervisée LDA, on va essayer d'extraire les topics cachés dans l'ensemble de corpus. Suivi par une analyse exploratoire des résultats

- Prédiction des tags : dans cette partie, on découvre la classification en multi-label. Le but serait de prédire l'ensemble des tags pour une question donnée. Avec des méthodes supervisées on va essayer d'optimiser nos métriques d'évaluations

Ce rapport présente une explication détaillée des différentes étapes de traitement des données textuelles et une étude des différents modèles testés. Enfin, nous expliquerons en quoi le modèle choisi répond le mieux à nos besoins.

Préparation des données :

Cette partie sera dédiée à la préparation de notre jeu de données finale :

- Récupération des données : à l'aide d'une requête SQL à partir de **Stack-Exchange Explorer**

- Nettoyage de l'ensemble des données

- Analyse exploratoire du corpus

Importation des données :

Stack Overflow propose un outil d'export de données qui est le "**Stackexchange Explorer**", qui recense un grand nombre de données authentiques de la plateforme d'entraide. A l'aide d'une requête SQL on va extraire les données nécessaires à l'entraînement de nos algorithmes.

On a sélectionné les questions avec leurs titres et leurs tags des postes qui sont publié dans la période compris entre 2019-01-01 et 2023-05-01.

```sql
select
  title,body,tags
from
  Posts
where
  CreationDate between '2019-01-01' and '2023-05-01' and len(tags)>0 ;
```

On a pu retirer en tout prêt de 50000 questions, qui nous semble suffisant pour notre étude.

	title	body	tags
0	Custom_Vision_Prediction_3.0 REST API Giving 4...	<p>I am trying to call Custom Vision Predictio...	<azure><azure-cognitive-services><microsoft-cu...
1	Cant get the query for AzureDiagnostics to work	<p>I cant get this query to work. I got this q...	<azure><azure-log-analytics>
2	Would it possible to use all memory of GPUs wi...	<p>There is a <code>model</code> and two GPUs....	<python><gpu><pytorch>
3	Command not found while using sudo [UBUNTU 19.04]	<p>When i try to run xampp server using the co...	<php><apache><ubuntu><xampp><lampp>
4	Get the prop and value of dynamically added in...	<p>There are two input fields in my form initi...	<react-native><redux><react-redux>

Nettoyage des données :

Nettoyage Générale :

Le premier nettoyage permet de standardiser le texte. Pour cela on met en minuscule tout le texte, on supprime les liens et les balises de code, on supprime les ponctuations (sauf le # pour conserver le langage c#), on supprime les chiffres, et on enlève les contractions (exemple : i'm en i am).

Suppression de catégories de mot :

Afin de garder l'information essentiel, les verbes, les adjectifs, les adverbes vont être supprimés.

Tokenisation :

Pour continuer le processus de nettoyage une tokenisation est nécessaire pour séparer les mots, tout en gardant le C et le # .

Lemmatisation :

Une fois les mots séparés, une lemmatisation (ou racinisation) est faite. Ce processus consiste à garder la racine des mots et donc garder uniquement le sens des mots.

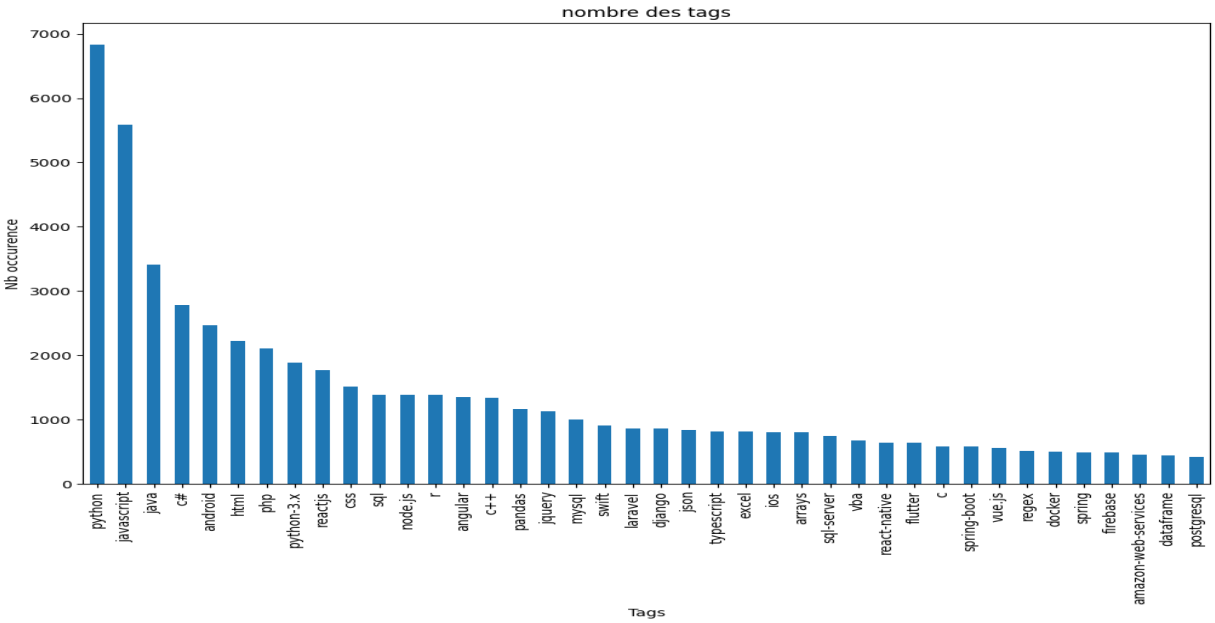
Nettoyage des Tags :

Les opérations suivantes ont pour but de préparer les tags en vue de leur utilisation ultérieure dans l'analyse ou le modèle de catégorisation des questions.

-Remplacement des caractères "<" par une chaîne vide
-Remplacement des caractères ">" par des virgules
-Suppression du dernier caractère (une virgule)

	title	body	tags
0	Custom_Vision_Prediction_3.0 REST API Giving 4...	<p>I am trying to call Custom Vision Predictio...	azure,azure-cognitive-services,microsoft-custo...
1	Cant get the query for AzureDiagnostics to work	<p>I cant get this query to work. I got this q...	azure,azure-log-analytics
2	Would it possible to use all memory of GPUs wi...	<p>There is a <code>model</code> and two GPUs....	python,gpu,pytorch

-Compter le nombre de mots différents dans la colonne Tags en utilisant un séparateur spécifié (une virgule).
-Création d'une data frame pour regrouper et compter les occurrences de chaque tag.
-Tri de le data frame selon le nombre d'occurrences de chaque tag.
-Sélection des 40 tags les plus fréquents
-Visualisation du nombre d'occurrences des tags les plus fréquents à l'aide d'un graphique à barres.



-Filtrage les données initiales pour ne conserver que celles ayant des tags.
-Remplacement des virgules par des espaces dans les tags
-Tokenisation des tags

	title	body	tags
2	memory gpus model	<p>There is a <code>model</code> and two GPUs....	[python]
3	ubuntu	<p>When i try to run xampp server using the co...	[php]
4	prop value inputs redux react	<p>There are two input fields in my form initi...	[react-native]
5	css js web application servlet	<p><a href="https://i.stack.imgur.com/k3Vr3.pn...	[javascript, java, css]
6	factors analysis array dataframe	<p>when doing the factor analysis with <code>f...	[python]

Nettoyage des titres :

-Conversion en minuscules : Les titres sont convertis en minuscules pour normaliser le texte et éviter les ambiguïtés entre les mots en majuscules et en minuscules.
-Suppression de la ponctuation : Toute ponctuation est supprimée des titres, à l'exception du symbole "#".
-Suppression des chiffres : Les chiffres et les mots contenant des chiffres sont supprimés des titres.
-Suppression des liens : Les liens URL sont supprimés des titres.
-Suppression des sauts de ligne : Les caractères de saut de ligne sont remplacés par un espace.
-Remplacement de certains mots : Certains mots couramment abrégés sont remplacés par leur forme complète (par exemple, "i'm" par "i am", "i've" par "i have").
-Suppression des espaces en trop : Les espaces en trop sont supprimés pour obtenir un texte plus propre.
-Remplacement de "c #" par "cSharp" : Les occurrences de "c #" sont remplacées par "cSharp" pour uniformiser la représentation du langage de programmation C#.
-Extraction des noms et pronoms : Les noms et pronoms sont extraits des titres à l'aide de l'analyse morphologique réalisée avec le modèle linguistique de spaCy.
-Tokenisation : Les titres sont divisés en tokens individuels.
-Lemmatisation : Chaque token dans les titres est ramené à sa forme de base (lemme).
-Suppression des mots vides (stop words) : Les mots courants qui n'apportent pas de signification particulière, tels que "and", "the", "but", sont supprimés des titres en utilisant une liste prédéfinie de mots vides de la langue anglaise.

```
2                                  [memory, gpus, model]
3                                             [ubuntu]
4                    [prop, value, input, redux, react]
5                    [cs, j, web, application, servlet]
6                  [factor, analysis, array, dataframe]
7                                       [member, list]
9       [count, value, field, value, query, case, stmt]
10                   [python, csv, bucket, import, csv]
11                        [code, column, result, table]
```

Nettoyage des questions:

Ces opérations ont pour but de nettoyer, normaliser et préparer les questions en vue de leur utilisation ultérieure dans l'analyse et la modélisation.
-Suppression des balises de code : Les balises de code HTML, représentées par <code>. Tout le contenu entre les balises de code est remplacé par une chaîne vide.
-Regroupement de "c" et "#" : Les occurrences de "#" sont regroupées avec le caractère précédent "c", transformant ainsi "c #" en "c#". Cela est fait pour normaliser la représentation du langage de programmation C#.
-Conversion en minuscules : Les titres sont convertis en minuscules pour normaliser le texte et éviter les ambiguïtés entre les mots en majuscules et en minuscules.
-Suppression de la ponctuation : Toute ponctuation est supprimée des titres, à l'exception du symbole "#".
-Suppression des chiffres : Les chiffres et les mots contenant des chiffres sont supprimés des titres.
-Suppression des liens : Les liens URL sont supprimés des titres.
-Suppression des sauts de ligne : Les caractères de saut de ligne sont remplacés par un espace.
-Remplacement de certains mots : Certains mots couramment abrégés sont remplacés par leur forme complète (par exemple, "i'm" par "i am", "i've" par "i have").
-Suppression des espaces en trop : Les espaces en trop sont supprimés pour obtenir un texte plus propre.
-Extraction des noms et pronoms : Les noms et pronoms sont extraits des questions.
-Tokenisation : Les questions sont divisées en tokens individuels
-Lemmatisation : Chaque token dans les questions est ramené à sa forme de base (lemme).
Suppression des mots vides (stop words) : Les mots courants qui n'apportent pas de signification particulière sont supprimés des questions en utilisant une liste prédéfinie de mots vides de la langue anglaise.

	title	body	tags
2	[memory, gpus, model]	[gpus, gpu, image, memory, error, memory, imag...	[python]
3	[ubuntu]	[xampp, server, command, xampp, command, result]	[php]
4	[prop, value, input, redux, react]	[input, field, form, button, field, redux, dif...	[react-native]
5	[cs, j, web, application, servlet]	[project, structure, intellij, idea, java, web...	[javascript, java, css]
6	[factor, analysis, array, dataframe]	[factor, analysis, lib, result, array, number,...	[python]

Les modélisations :

Une fois le texte nettoyé et contenant uniquement l'information utile, il faut trouver le bon modèle afin de prédire les tags associés aux questions.
Pour cela il faut transformer le texte en structure compréhensible par le système de modélisation.
Et ensuite il faut tester différents modèles afin de trouver le plus efficace.

Représentation du corpus :

Bag of Words :

La méthode bag of words permet de créer un vecteur du dictionnaire du corpus, en indiquant le nombre de fois que chaque mot apparait dans le texte.

Exemple de bag of words

Texte ->

```
[past, microsoft, web, application, stress, tool, pylot,
stress, test, web, application, home, page, login,
script, site, walkthrough, ecommerce, site, item, cart,
checkout, homepage, handful, developer, problem,
scalability, problem, stage, launch, url, tool,
microsoft, homer, microsoft, web, application, stress,
tool, pylot, report, tool, sense, hour, kind, load,
site, bug, bottleneck, instance, web, server,
misconfigurations, tool, success, approach, part, kind,
formula, number, user, app, number, stress, test,
application, stress, test, web, application]
```

Bag Of Words

```
[[('app', 1),
  ('application', 5),
  ('approach', 1),
  ('bottleneck', 1),
  ('bug', 1),
  ('cart', 1),
  ('checkout', 1),
  ('developer', 1),
  ('formula', 1),
  ('handful', 1),
  ('home', 1),
  ('homepage', 1),
  ('hour', 1),
  ('instance', 1),
  ('item', 1),
  ('kind', 2)…]]
```

TF IDF (Term Frequency Inverse Document Frequency) :

$$TFIDF_{t,d,D} = TF_{t,d} \times IDF_{t,D}$$

- $TFIDF_{t,d,D}$: Importance d'un terme t dans un document d
- $TF_{t,d}$: Fréquence d'un terme t dans un document d
- $IDF_{t,D}$: Importance du terme t dans l'ensemble des documents D

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Texte

[past, microsoft, web, application, stress, tool, pylot,
stress, test, web, application, home, page, login,
script, site, walkthrough, ecommerce, site, item, cart,
checkout, homepage, handful, developer, problem,
scalability, problem, stage, launch, url, tool,
microsoft, homer, microsoft, web, application, stress,
tool, pylot, report, tool, sense, hour, kind, load,
site, bug, bottleneck, instance, web, server,
misconfigurations, tool, success, approach, part, kind,
formula, number, user, app, number, stress, test,
application, stress, test, web, application]

Exemple de TF IDF

```
access  action  activity  address  advantage  algorithm  alternative  \
   0.0     0.0       0.0      0.0        0.0        0.0          0.0

amount  android  angularjs  answer  api       app  application  approach  \
   0.0      0.0        0.0     0.0  0.0  0.081998     0.381566  0.103392

apps  area  argument  array  article  aspnet  attempt  attribute  \
 0.0   0.0       0.0    0.0      0.0     0.0      0.0        0.0

background  bar  base  bash  behavior  behaviour  benefit  bit  block  \
       0.0  0.0   0.0   0.0       0.0        0.0      0.0  0.0    0.0

book  bootstrap  bottom  box  branch  browser       bug  build  builtin  \
 0.0        0.0     0.0  0.0     0.0      0.0  0.117281    0.0      0.0
```

Modèles d'apprentissage :

LDA (Latent Dirichlet Allocation) :